Performance and Portability Lessons from HACC

Hal Finkel, **Nick Frontiere**,
Katrin Heitmann, Joe Insley,
Vitali Morozov, Tom Peterka,
Adrian Pope, Venkat Vishwanath
Argonne
National Laboratory

Zarija Lukic
Lawrence Berkeley
National Laboratory

David Daniel, Patricia Fasel
Los Alamos
National Laboratory

Salman Habib
HEP and MCS Divisions
Argonne National Laboratory

Computation Institute
Argonne National Laboratory
University of Chicago

Kavli Institute for Cosmological Physics
University of Chicago

ATPESC
July 31, 2013
Not a talk for software developers!

Performance and Portability I

- **Performance (assuming you are solving a new problem, not doing 'ports')**
  - Are you sure you want brute speed? (There is always a price -- realize all HPC machines are poorly balanced)
  - Or do you just want to run a 'large' problem with acceptable time to solution? (This is the general case)
  - **Step I: Know** what you want, if performance is a priority it must be designed in right at the start, you'll never get it afterwards (optimizing gains are often minimal to non-existent)
  - **Step II:** If performance is needed, make sure you understand the global science problem(s) being addressed; you may have to start from scratch! **There's no replacement for domain knowledge**
  - Factor of two rule -- given human constraints (and Moore's law), it is not usually worth it to go for the last factor of two, but there are exceptions -- HACC is one
  - **Step III:** Obtaining performance is painful, so design for the future -- what can you rely on, what can disappear, what can change, what can break -- the more parameters you can control, the better -- HPC systems are not your laptop: **Learn from experience**
  - General Advice (mostly obvious): On-chip/node optimization comes first**,** minimize number of performance 'hot spots' to the extent possible, ditto with data motion (aim to be compute-bound, avoid look-ups), avoid forest/tree syndromes, think about sacrificing memory for speed wherever possible, vectorize everything, FMAs are your friends, talk to performance gurus, do not resort to assembly unless desperate, etc. etc.

Performance and Portability II

- **Portability (assuming you are developing new code)**
  - Three scales of code development: individual (‘idiosyncratic’), small team (‘hot shots’), big team to open source (‘industrial’)
  - Compute environment: small-scale (‘individual PI’, low diversity hardware), medium-scale (‘single project’, somewhat diverse hardware), large-scale (‘multiple projects’, very diverse hardware) -- **note scale here does not refer to problem size!**
  - **Step I:** Consider which categories your situation falls into, this will help set the portability constraints
  - Concrete advice is difficult; situations vary, look around you and see what other people are doing -- learn from them (adopt/reuse what works, dump what does not, be ruthless)
  - Simplicity is good (learn from Google!), avoid nonfunctional ‘adornments’
  - Design for the future -- software life cycles should be long, **but often are not**
  - **Step II:** Most science projects start with a compact ‘software core’ that grows in multiple directions, pay attention to planning the structure of the core and the extension paths -- things will often not work as expected so make sure the structure is sufficiently flexible -- starting from scratch should be largely a reconfiguration of key software elements; identify these elements and design around them
  - Performance and portability are often in opposition, but they can be co-aligned -- as in HACC

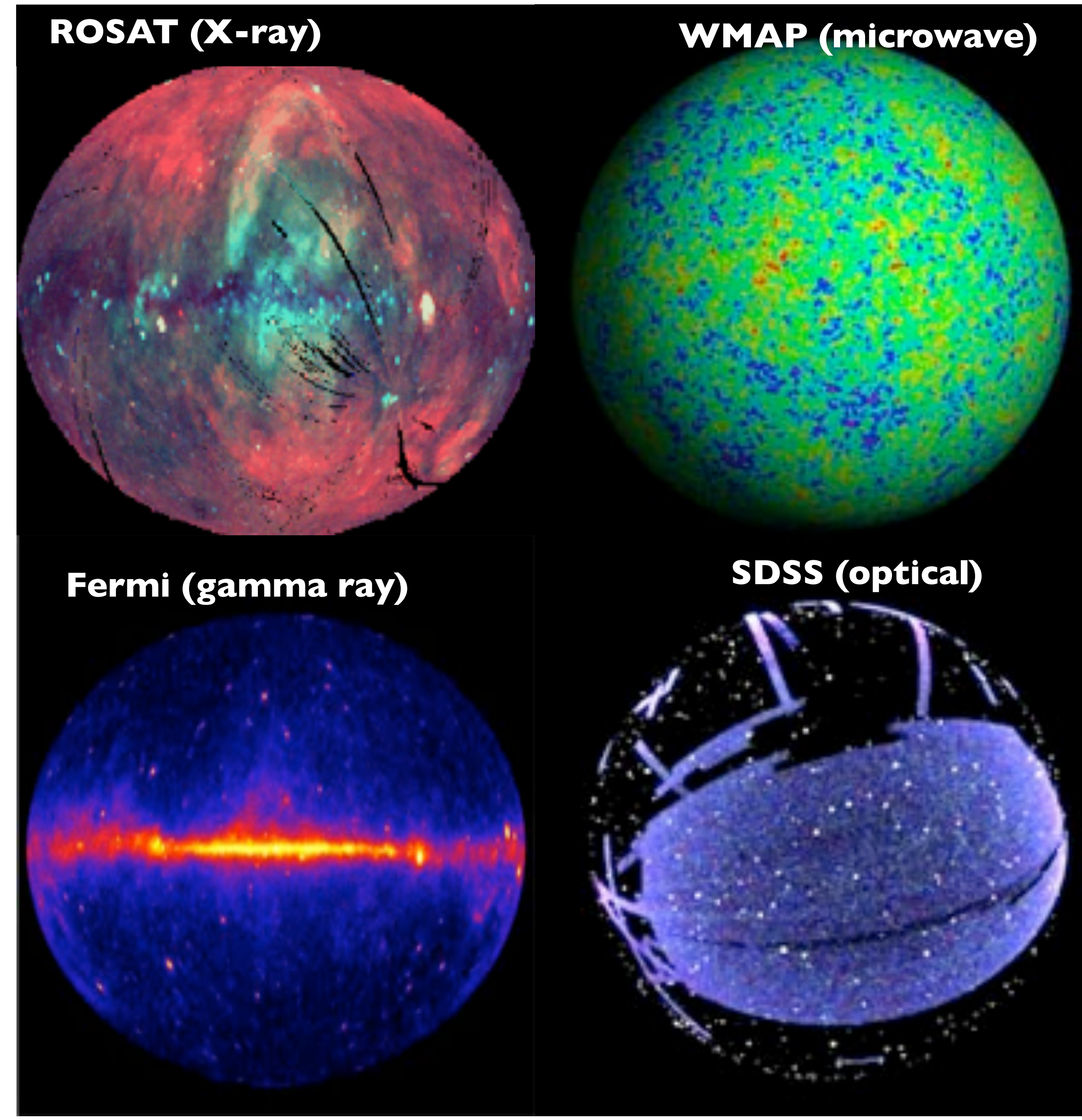
What is HACC?

HACC (Hardware/Hybrid Accelerated Cosmology Code) Framework

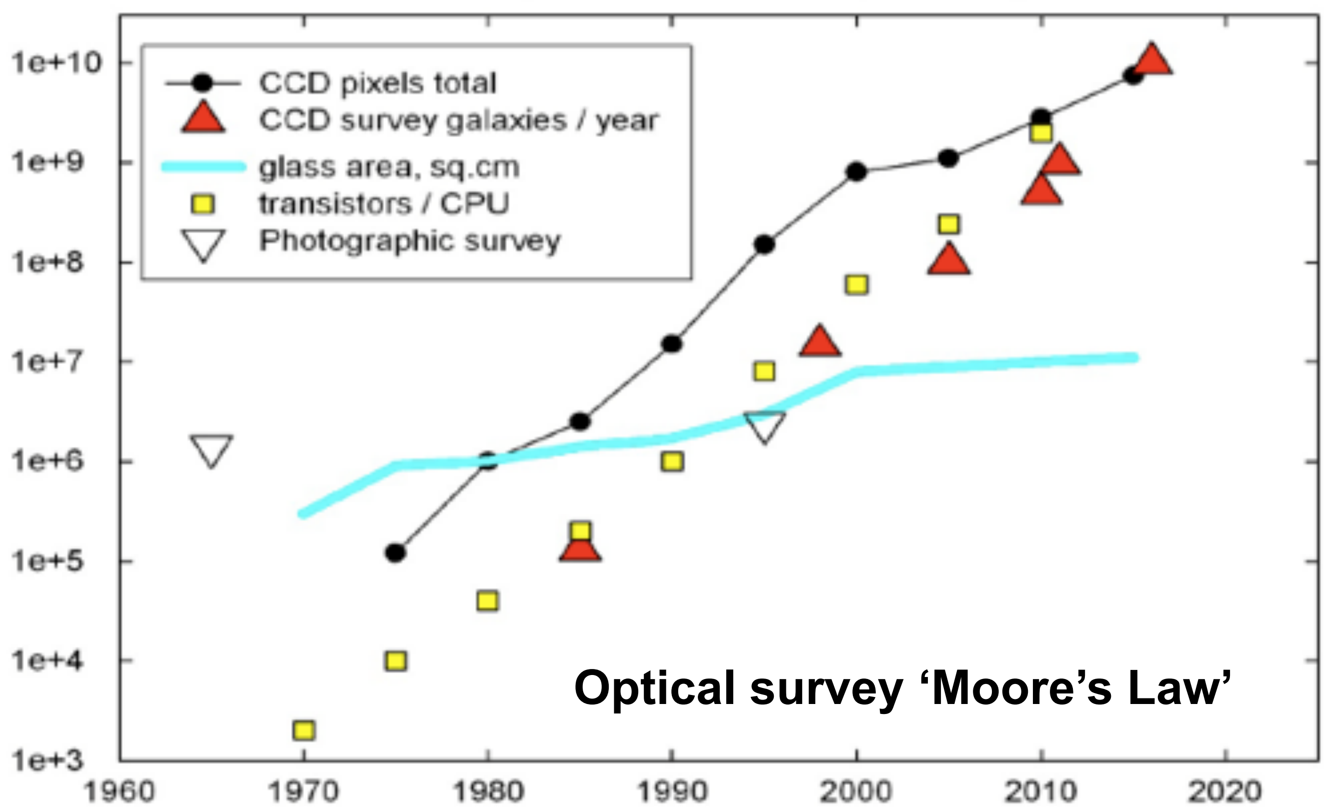
- **HACC does very large cosmological simulations**
  - Design Imperative: Must run at high performance on all supercomputer architectures at full scale
  - Highest performance ever achieved on the BG/Q by a science code
  - Combines a number of algorithms using a ‘mix and match’ approach
  - Perfect weak scaling
  - Strong scales to better than 100 MB/core
  - Currently running the world’s largest cosmology simulation on Mira

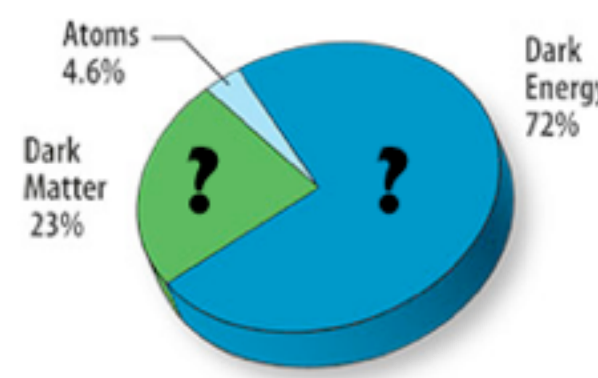
Why HACC I?: ‘Precision’ Cosmology

- Instrumentation Advances
- Cosmic Acceleration
- Nature of Dark Matter
- Primordial Fluctuations
- Neutrinos
- Cosmic Structure Formation



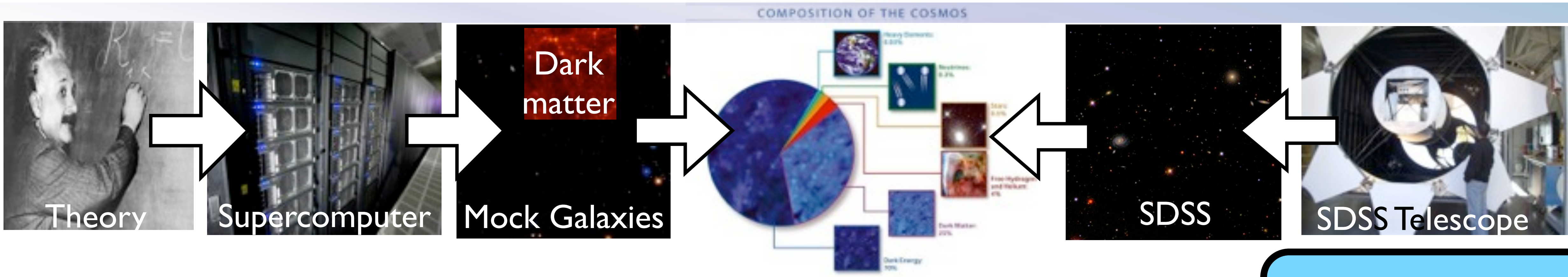
The Source of Knowledge: Sky Surveys



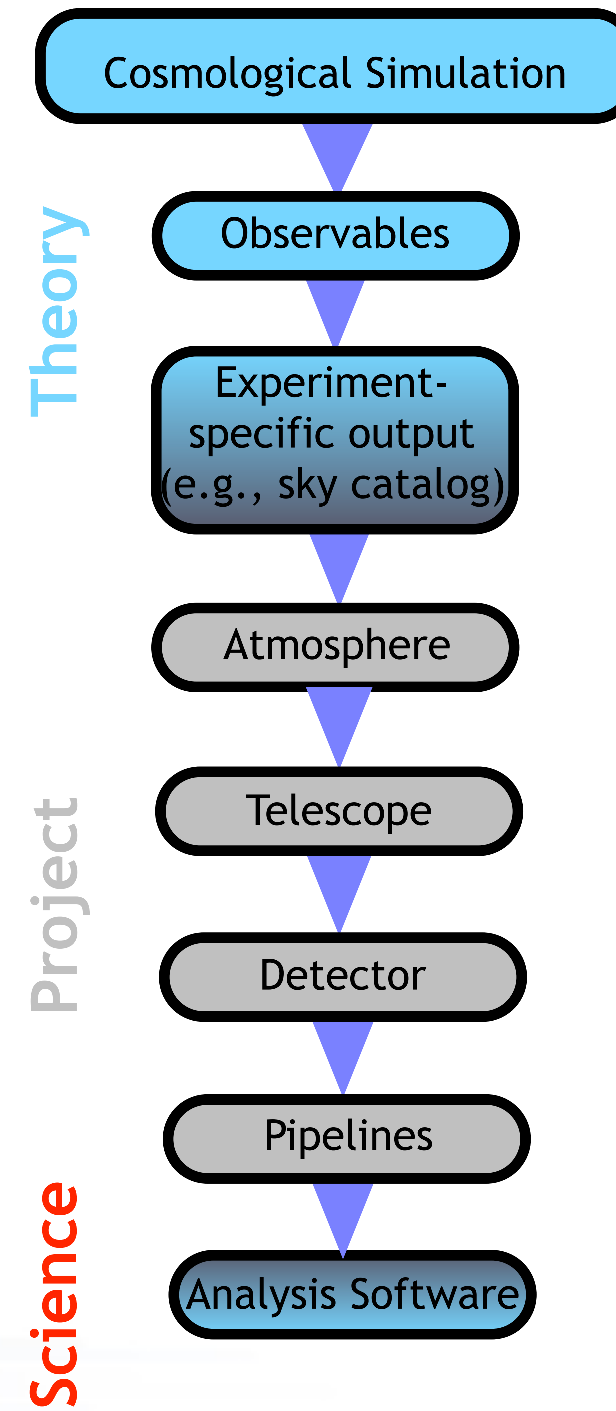


The Cosmic Puzzle: Who ordered the rest of it?

Why HACC 2?: Key Role of Computation



- **Three Roles of Cosmological Simulations**
  - Basic theory of cosmological probes
  - Production of high-fidelity ‘mock skys’ for end-to-end tests of the observation/analysis chain
  - Essential component of analysis toolkits
- **Extreme Simulation and Analysis Challenges**
  - Large dynamic range simulations; control of subgrid modeling and feedback mechanisms
  - Design and implementation of complex analyses on large datasets; new fast (approximate) algorithms
  - Solution of large statistical inverse problems of scientific inference (many parameters, ~10-100) at the ~1% level



Simulating the Universe

- **Key Role of Gravity**
  - Gravity dominates at large scales: solve the Vlasov-Poisson equation (VPE)
  - VPE is 6-D and cannot be solved as a PDE
- **N-Body Methods**
  - No shielding in gravity (essentially long range interactions)
  - Technique is naturally Lagrangian
  - Are errors controllable?
- **More Physics**
  - Smaller scale 'gastrophysics' effects added via subgrid modeling or post-pocessing (major topic)
- **Phenomenology**
  - Calibrate simulations against observations

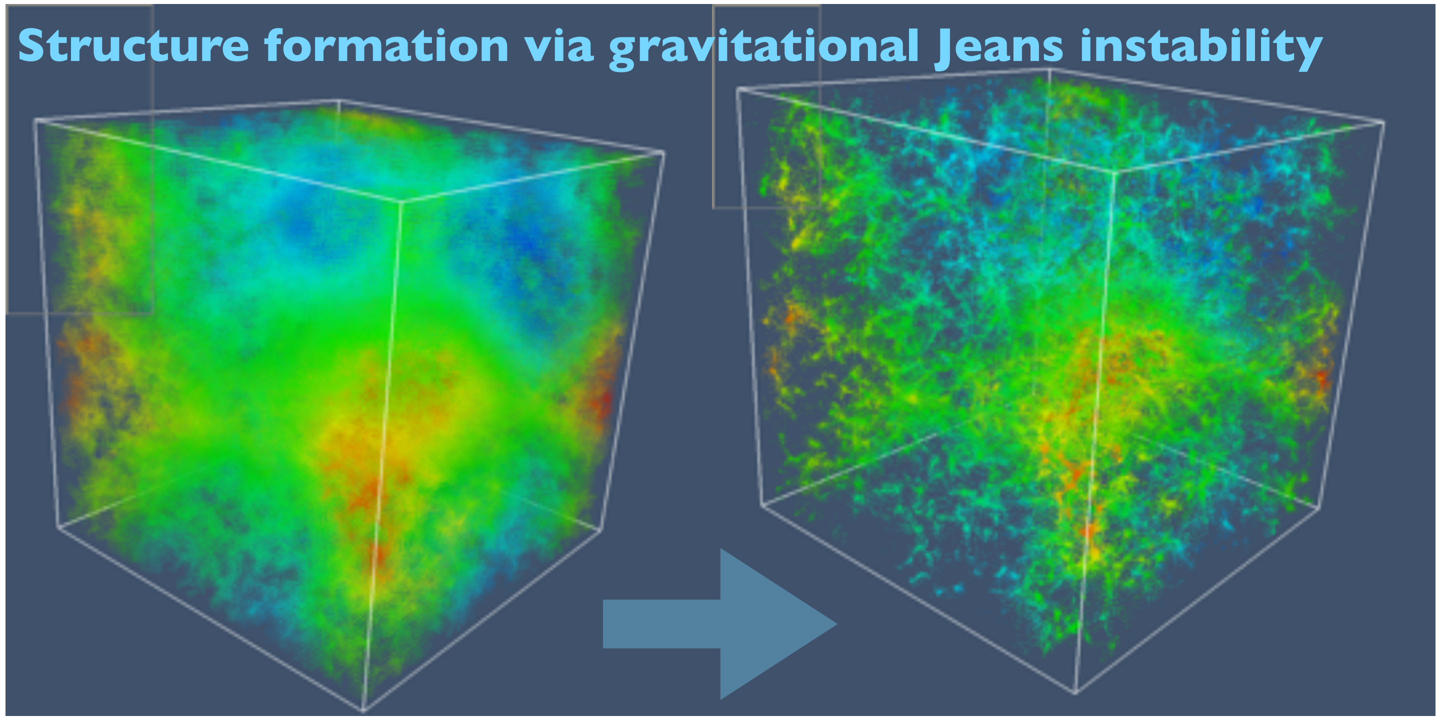
$$\frac{\partial f_i}{\partial t} + \dot{\mathbf{x}}\frac{\partial f_i}{\partial \mathbf{x}} - \nabla\phi\frac{\partial f_i}{\partial \mathbf{p}} = 0, \qquad \mathbf{p} = a^2\dot{\mathbf{x}},$$

$$\nabla^2\phi = 4\pi G a^2(\rho(\mathbf{x},t) - \langle\rho_{\mathrm{dm}}(t)\rangle) = 4\pi G a^2\Omega_{\mathrm{dm}}\delta_{\mathrm{dm}}\rho_{\mathrm{cr}},$$

$$\delta_{\mathrm{dm}}(\mathbf{x},t) = (\rho_{\mathrm{dm}} - \langle\rho_{\mathrm{dm}}\rangle)/\langle\rho_{\mathrm{dm}}\rangle),$$

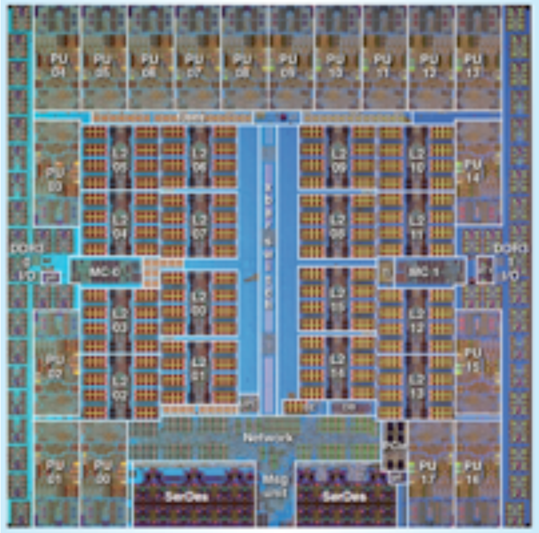
$$\rho_{\mathrm{dm}}(\mathbf{x},t) = a^{-3}\sum_i m_i\int d^3\mathbf{p}\, f_i(\mathbf{x},\dot{\mathbf{x}},t).$$

Cosmological Vlasov-Poisson Equation: **A 'wrong-sign' electrostatic plasma with time-dependent particle 'charge', Newtonian limit of the Vlasov-Einstein equations**

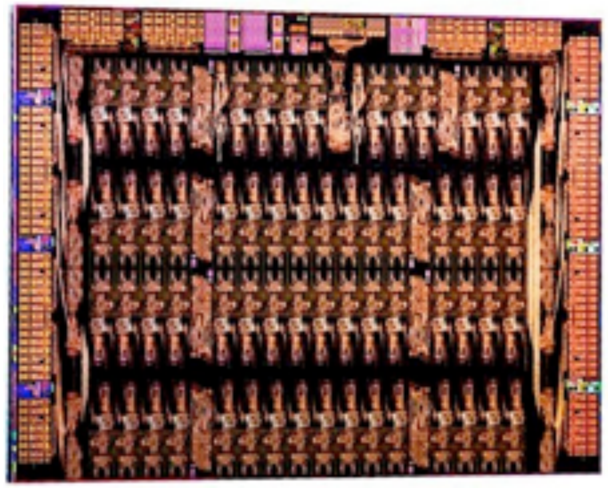


The N-Body Problem: Central Issues

- **Algorithms**
  - Naive P-P hopeless
  - Particle-Mesh: solve Poisson equation on a grid, interpolate forces onto particles, has limited resolution, but fast
  - Tree Codes: overcome resolution problem, but not efficient at long range, given required error properties
  - Hybrid codes: Combine above methods as needed (TPM, P3M, etc.)
  - Time-stepping: Multi-level schemes/locally adaptive
- **Parallel Implementations**
  - Performance and Scalability
  - Portability
- **Next-Generation Architectures ('Pile of PCs' to 'Pile of Cell Phones'?)**
  - Complex heterogeneous nodes (including power management)
  - Simpler cores, lower memory/core
  - Programming environments unclear



BQC:
- 16 cores
- 205 GFlops, 16 GB
- 32 MB L2, crossbar at 400 GB/s (memory connection is 40 GB/s)
- 5-D torus at 40 GB/s



Xeon Phi:
- 60 cores
- 1 TFlops, 8 GB
- 32 MB L2, ring at 300 GB/s (connects to cores/memory)
- 8 GB/s to host CPU

HACC's Domain: The 'Bleeding Edge'

- **Recall: Cosmology = Physics + Statistics**
  - Mapping the sky with large-area surveys
  - LSST: ~4 billion galaxies total; ~200,000 galaxies per sq. deg. or ~40K galaxies over a sky patch the size of the moon
  - To 'understand' a dataset this large (~100 PB), we need to model the distribution of matter down to the scales of the individual galaxies, and over the size of the entire survey: ~trillion particle simulations
- **Resolution:**
  - Force dynamic range greater than a million to one
  - Local overdensity variation is ~million to one
- **Computing 'Boundary Conditions':**
  - Total memory in the PB+ class
  - Performance in the 10 PFlops+ class
  - Wall-clock of ~days/week, in situ analysis

Can the entire observable Universe be 'stuffed' inside a supercomputer?

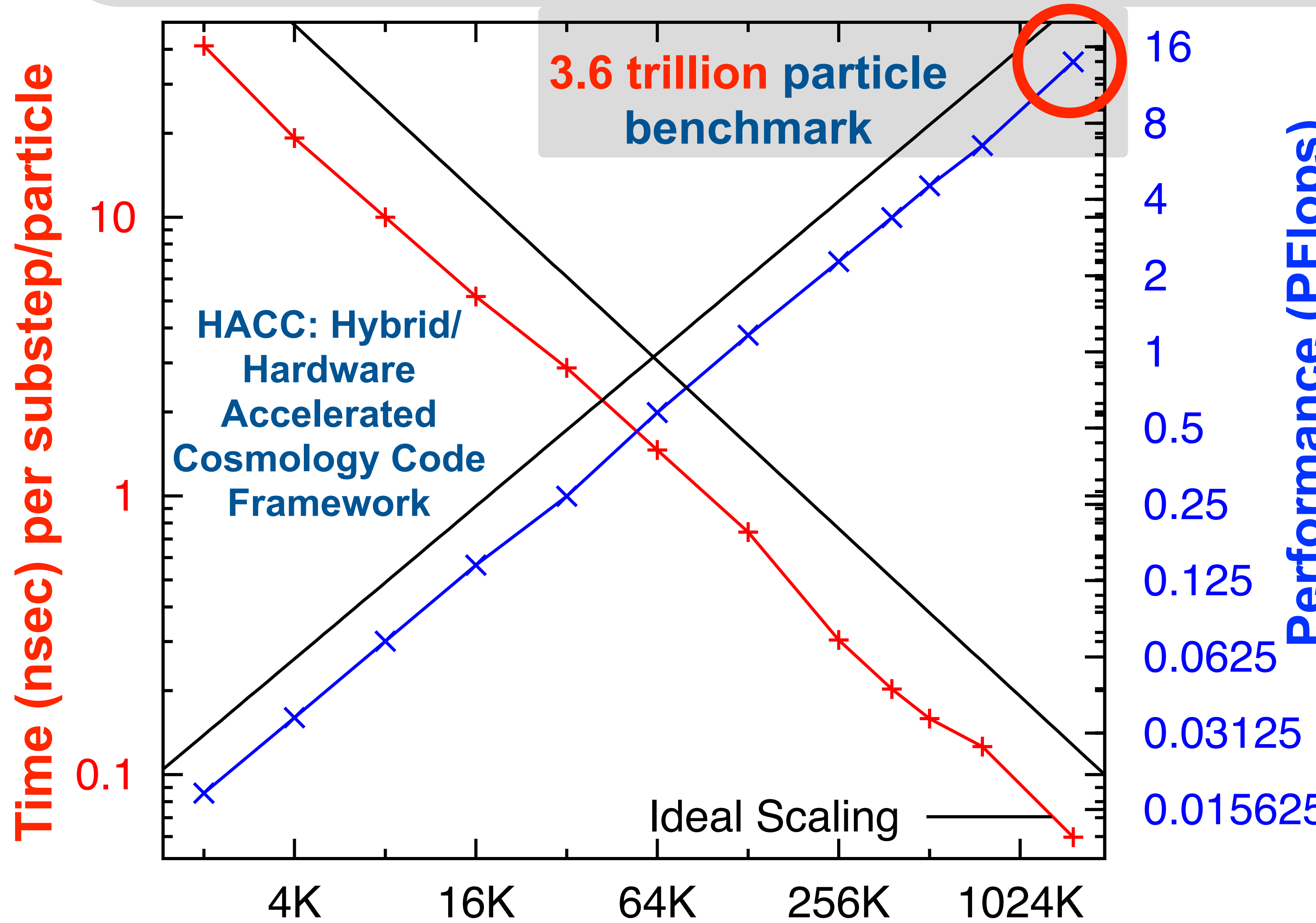
Can the Universe be run as a short computational 'experiment'?

Meeting the Challenge: HACC on BG/Q and CPU/GPU Systems

Cosmological N-Body Framework

- Designed for extreme performance AND portability, including heterogeneous systems
- Supports multiple programming models
- In situ analysis framework

13.94 PFlops, 69.2% peak, 90% parallel efficiency on 1,572,864 cores/MPI ranks, 6.3M-way concurrency

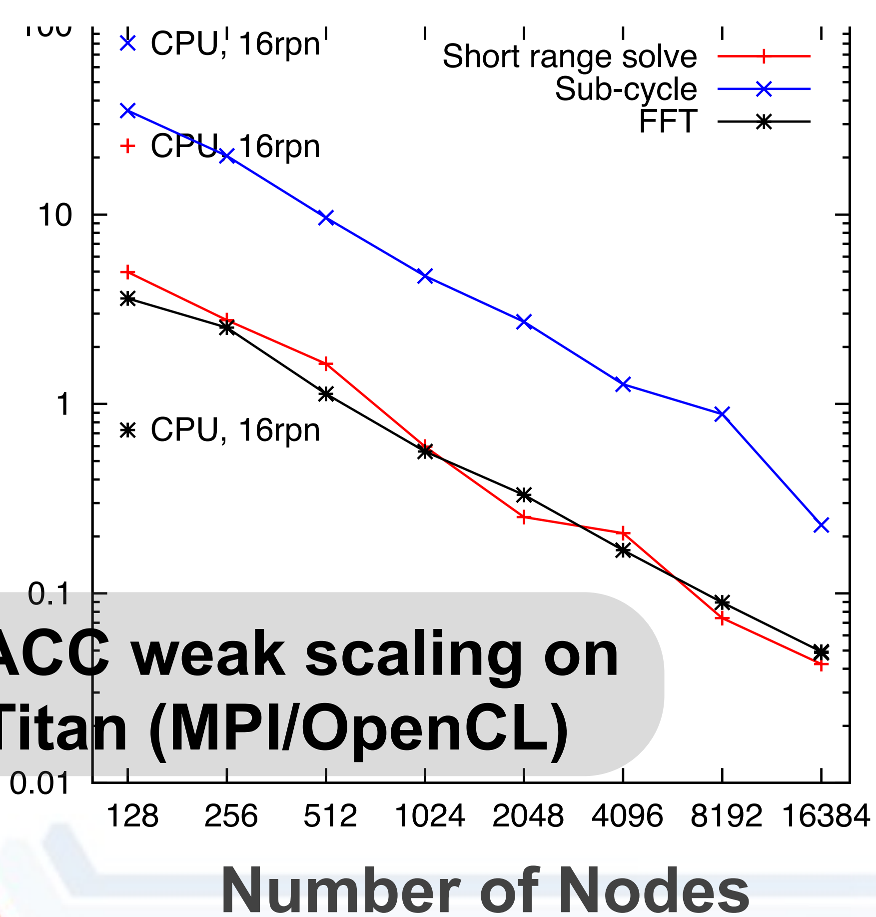


Number of Cores

Habib et al. 2012

Gordon Bell Award Finalist 2012, 2013

HACC weak scaling on the IBM BG/Q (MPI/OpenMP)



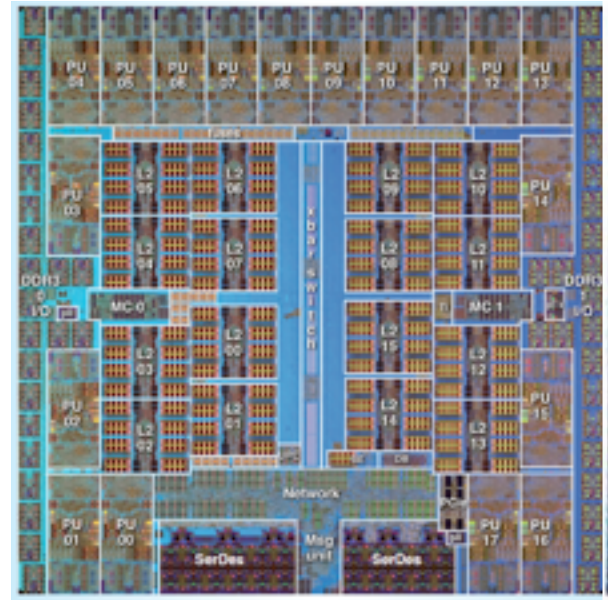
HACC weak scaling on Titan (MPI/OpenCL)

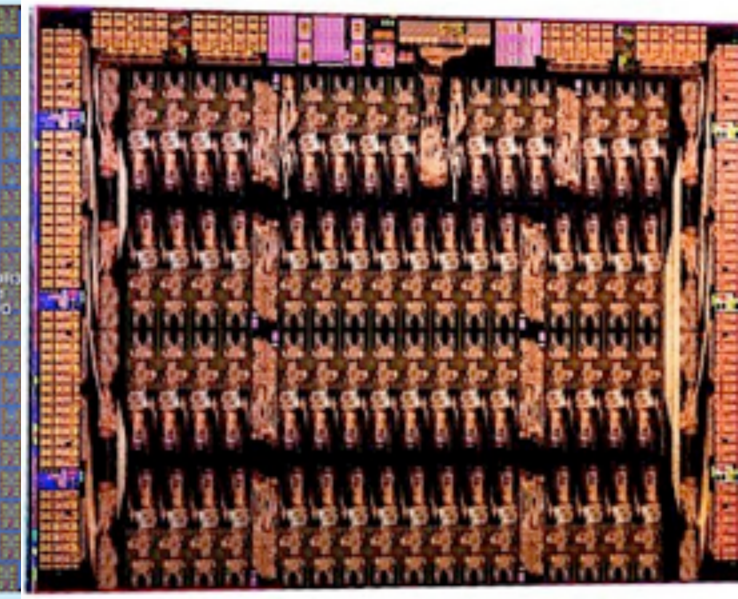
Co-Design vs. Code Design

- **HPC Myths**
  - The magic compiler
  - The magic programming model/ language (DSL)
  - Special-purpose hardware
  - Co-Design?
- **Dealing with (Current) HPC Reality**
  - Follow the architecture
  - Know the boundary conditions
  - There is no such thing as a ‘code port’
  - Think out of the box
  - Get the best team
  - Work together

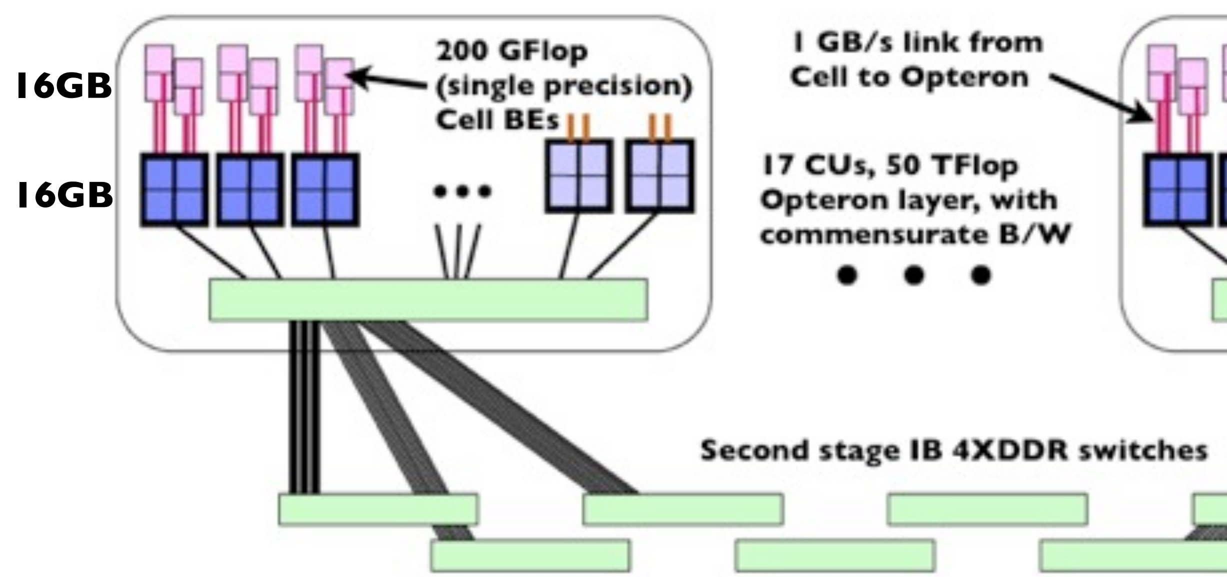
BQC:
- 16 cores
- 205 GFlops, 16 GB
- 32 MB L2, crossbar at 400 GB/s (memory connection is 40 GB/s)
- 5-D torus at 40 GB/s

Xeon Phi:
- 60 cores
- 1 TFlops, 8 GB
- 32 MB L2, ring at 300 GB/s (connects to cores and memory)
- 8 GB/s to host CPU





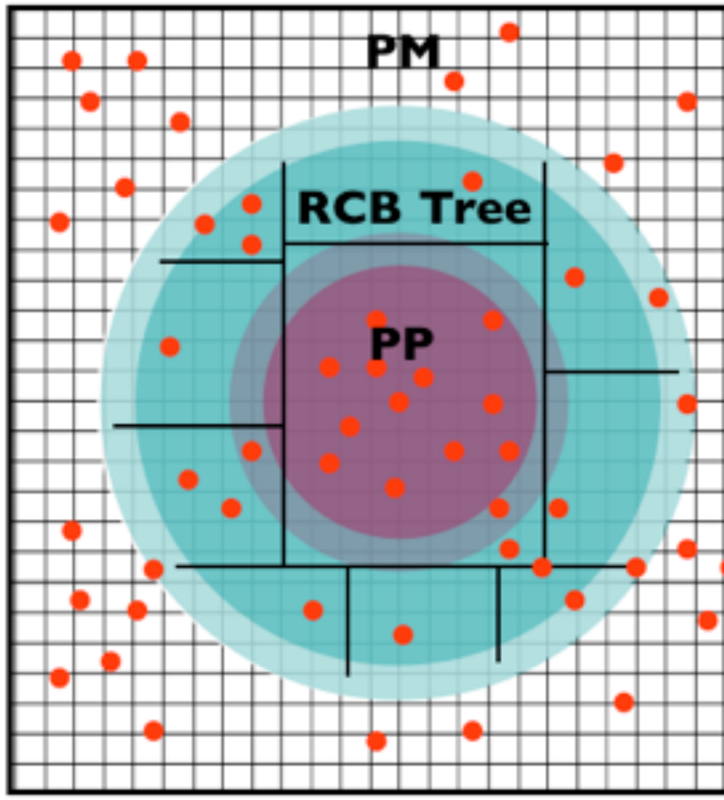
Average performance speed-up on ~10 applications codes on Titan is ~2 (ranging from 1.few to 7), but of Titan’s 27 PFlops, only 2.5 PFlops are in the CPU! What is wrong with this picture?



Roadrunner: The Original Driver for HACC

Opening the HACC ‘Black Box’: Design Principles

- **Optimize Next-Generation Code ‘Ecology’:** Numerical methods, algorithms, mixed precision, data locality, scalability, I/O, in situ analysis -- life-cycle significantly longer than architecture timescales
- **Framework design:** Support a ‘universal’ top layer + ‘plug-in’ optimized node-level components; minimize data structure complexity and data motion -- support multiple programming models
- **Performance:** Optimization stresses scalability, low memory overhead, and platform flexibility; assume ‘on your own’ for software support, but hook into tools as available (e.g., ESSL FFT)
- **Optimal Splitting of Gravitational Forces:** Spectral Particle-Mesh melded with direct and RCB tree force solvers, short hand-over scale (dynamic range splitting ~ 10,000 X 100)
- **Compute to Communication balance:** Particle Overloading
- **Time-Stepping:** Symplectic, sub-cycled, locally adaptive
- **Force Kernel:** Highly optimized force kernel takes up large fraction of compute time, no look-ups due to short hand-over scale
- **Production Readiness:** runs on all supercomputer architectures; exascale ready!



HACC force hierarchy (PPTreePM)

Splitting the Force: The Long-Range Solver

$$G_6(\mathbf{k}) = \frac{45}{128}\Delta^2 \left[\sum_i \cos\left(\frac{2\pi k_i \Delta}{L}\right) - \frac{5}{64}\sum_i \cos\left(\frac{4\pi k_i \Delta}{L}\right) + \frac{1}{1024}\sum_i \cos\left(\frac{8\pi k_i \Delta}{L}\right) - \frac{2835}{1024}\right]^{-1}$$

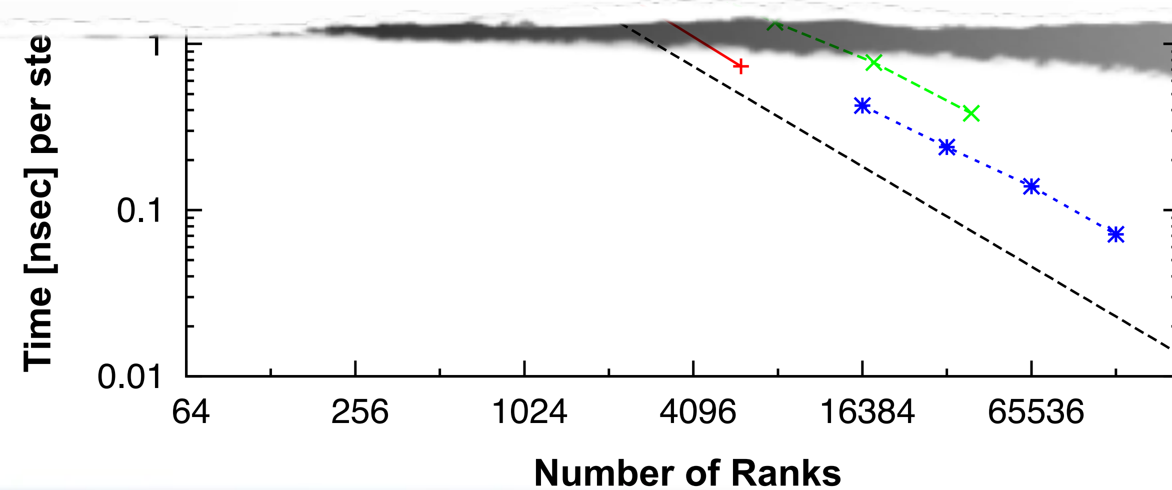
$$\left.\frac{\Delta f}{\Delta x}\right|_4 = \frac{4}{3}\sum_{j=-N+1}^{N} iC_j \mathrm{e}^{(2\pi j x/L)} \frac{2\pi j \Delta}{L}\frac{\sin(2\pi j\Delta/L)}{2\pi j \Delta/L} - \frac{1}{6}\sum_{j=-N+1}^{N} iC_j \mathrm{e}^{(2\pi j x/L)} \frac{2\pi j \Delta}{L}\frac{\sin(4\pi j\Delta/L)}{2\pi j \Delta/L}$$

where the C_j are the coefficients in the Fourier expansion of f

$$S(k) = \exp\left(-\frac{1}{4}k^2\sigma^2\right)\left[\left(\frac{2k}{\Delta}\right)\sin\left(\frac{k\Delta}{2}\right)\right]^{n_s}$$

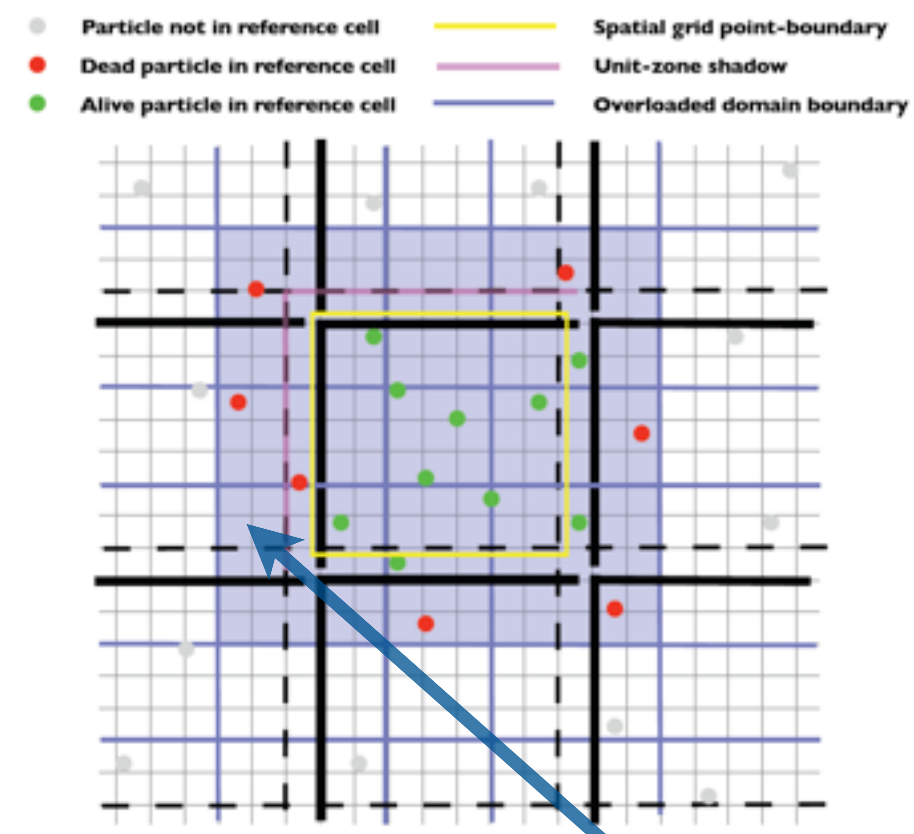
$$f_{grid}(r) = \frac{1}{r^2}\tanh(br) - \frac{b}{r}\frac{1}{\cosh^2(br)} + cr\left(1+dr^2\right)\exp\left(-dr^2\right) + e\left(1+fr^2+gr^4+lr^6\right)\exp\left(-hr^2\right)$$

- **Time-stepping uses Symplectic Sub-cycling:** Time-stepping via 2nd-order accurate symplectic maps with 'KSK' for the global timestep, where 'S' is split into multiple 'SKS' local force steps

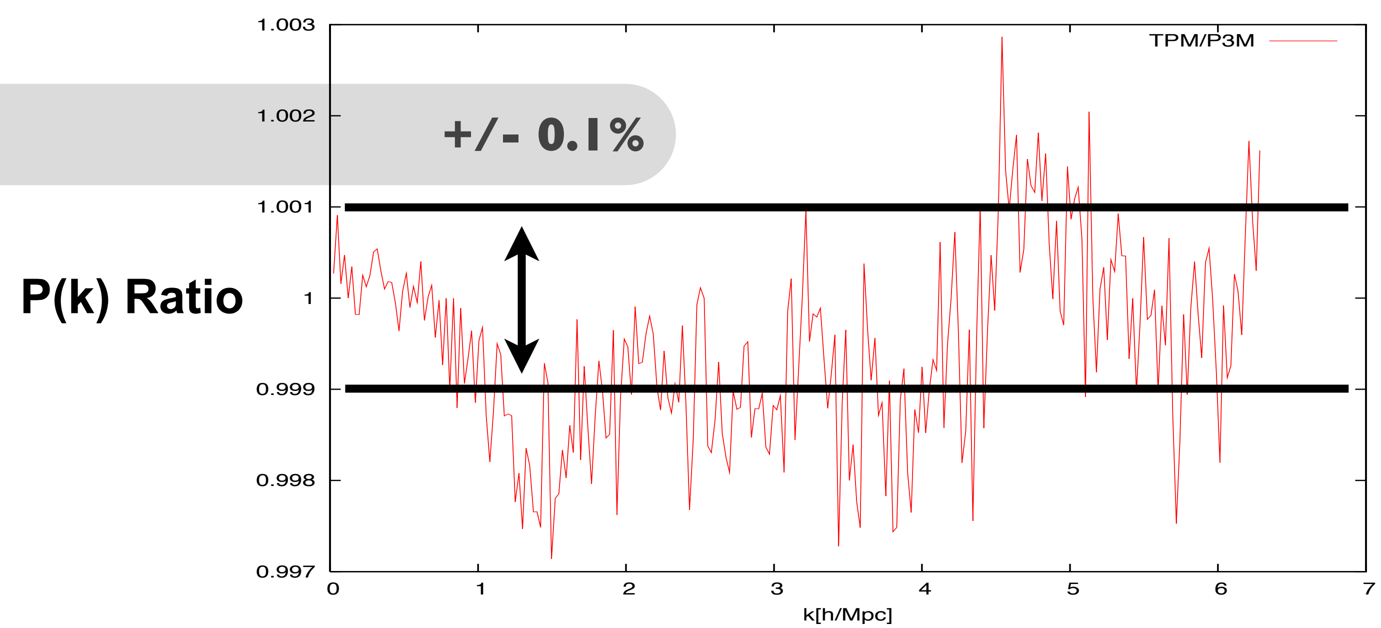


Particle Overloading and Short-Range Solvers

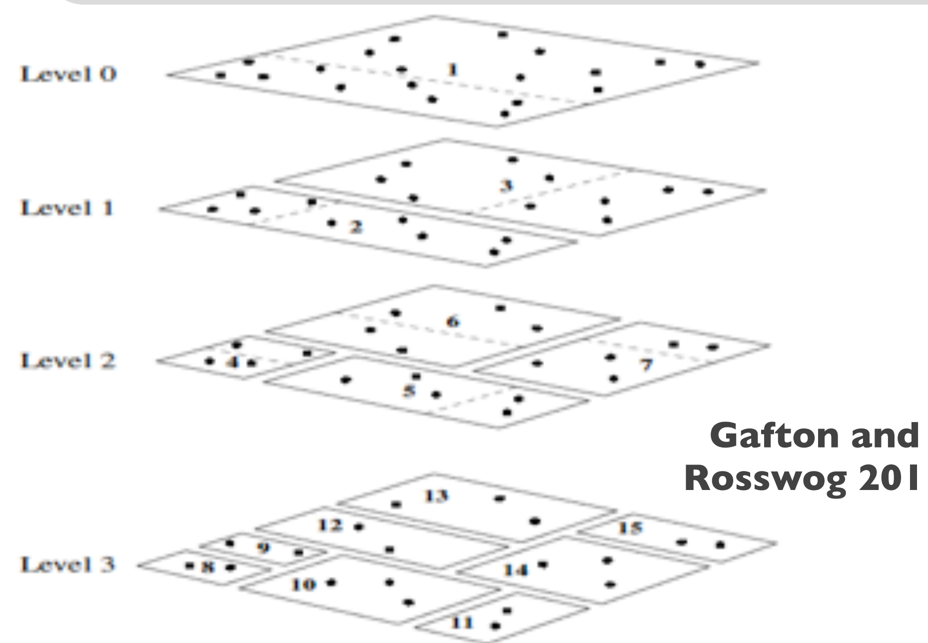
- **Particle Overloading:** Particle replication instead of conventional guard zones with 3-D domain decomposition -- minimizes inter-processor communication and allows for swappable short-range solvers **(IMPORTANT)**
- **Short-range Force:** Depending on node architecture switch between P3M and PPTreePM algorithms (pseudo-particle method goes beyond monopole order), by tuning number of particles in leaf nodes and error control criteria, optimize for computational efficiency
- **Error tests:** Can directly compare different short-range solver algorithms



Overload Zone (particle 'cache')



HACC Force Algorithm Test: PPTreePM vs. P3M



RCB Tree Hierarchy

Next Two Talks

- **Hal Finkel:** Short-Range Solver Performance (BG/Q; CPU)
- **Nick Frontiere:** GPU Issues (CPU/GPU Systems)